Best Practices, Process

Nathaniel Osgood

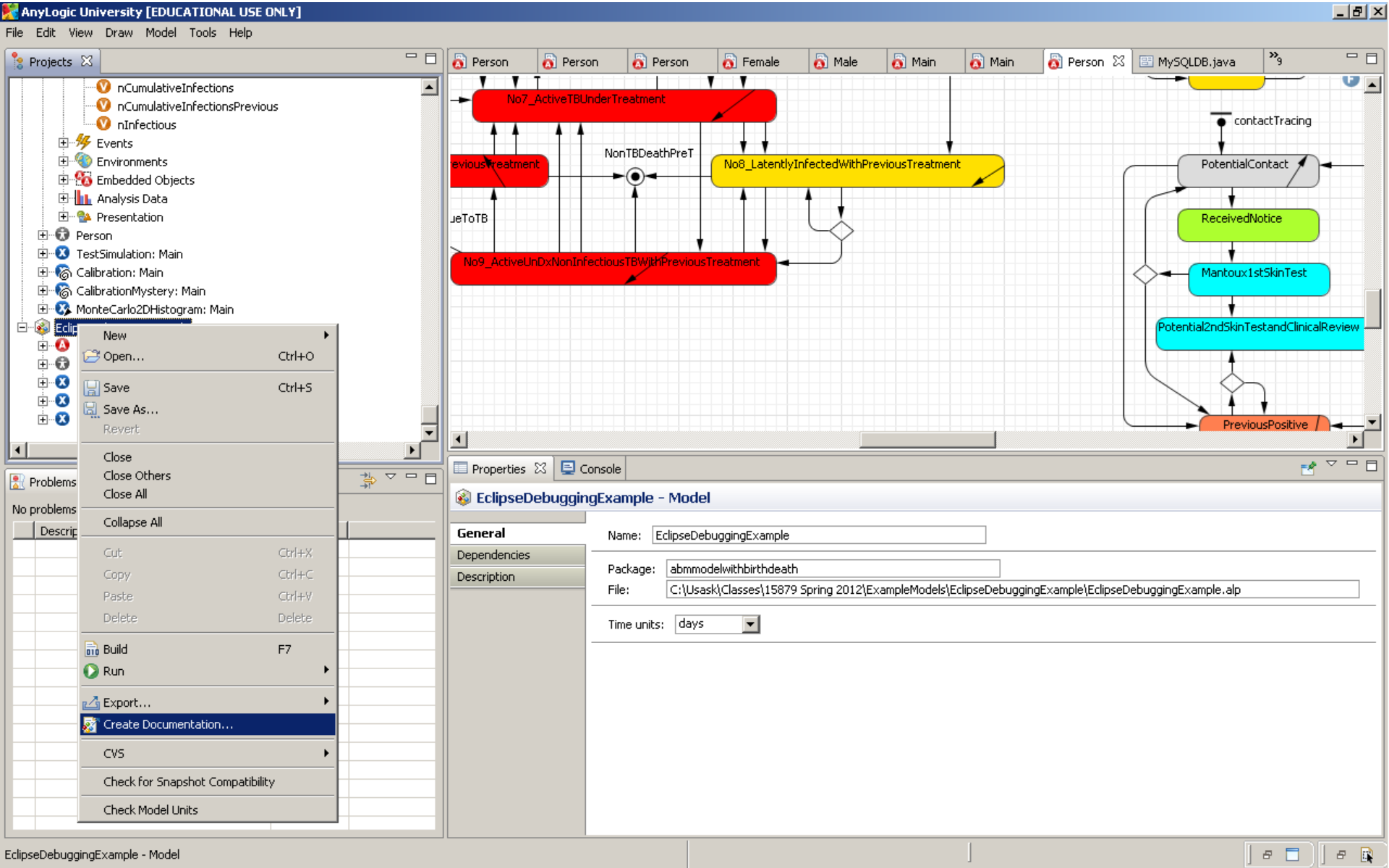
MIT 15.879

May 16, 2012

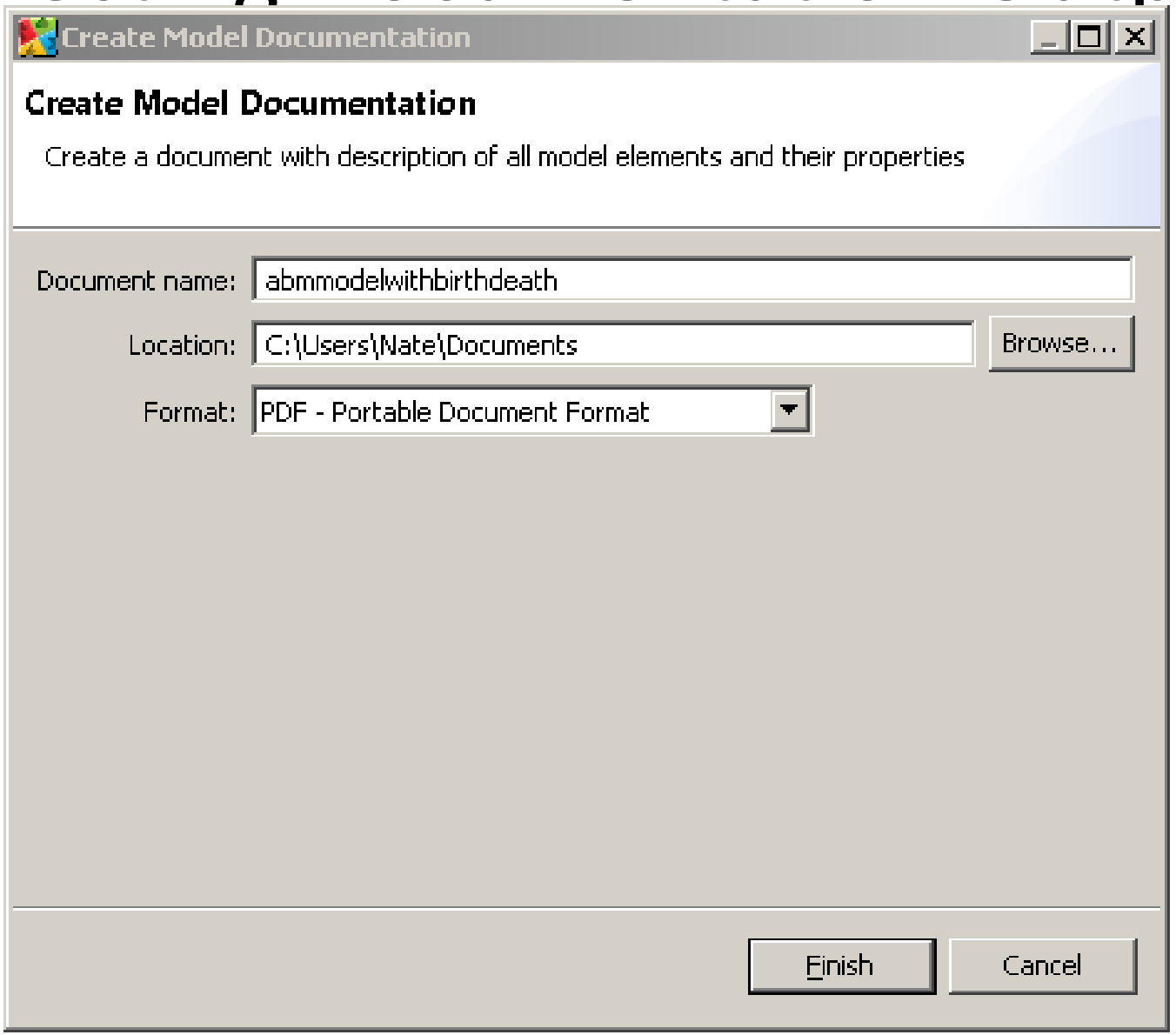
Recall: Process Suggestions

- Use discovery of bugs & oversights to find opportunities to improve Q & A and broader modeling process
- Use peer reviews (& especially inspections) to review
  - Preliminary design/Code/Tests
- Use tools for version control & documentation & referential integrity
  - Rigorous versioning
  - Document linkages between artifacts
- Keep careful track of experiments
- Strive for ongoing process improvement
- Use focused prototypes where appropriate
- Perform simple tests to verify functionality
- Integrate with others' work frequently & in small steps

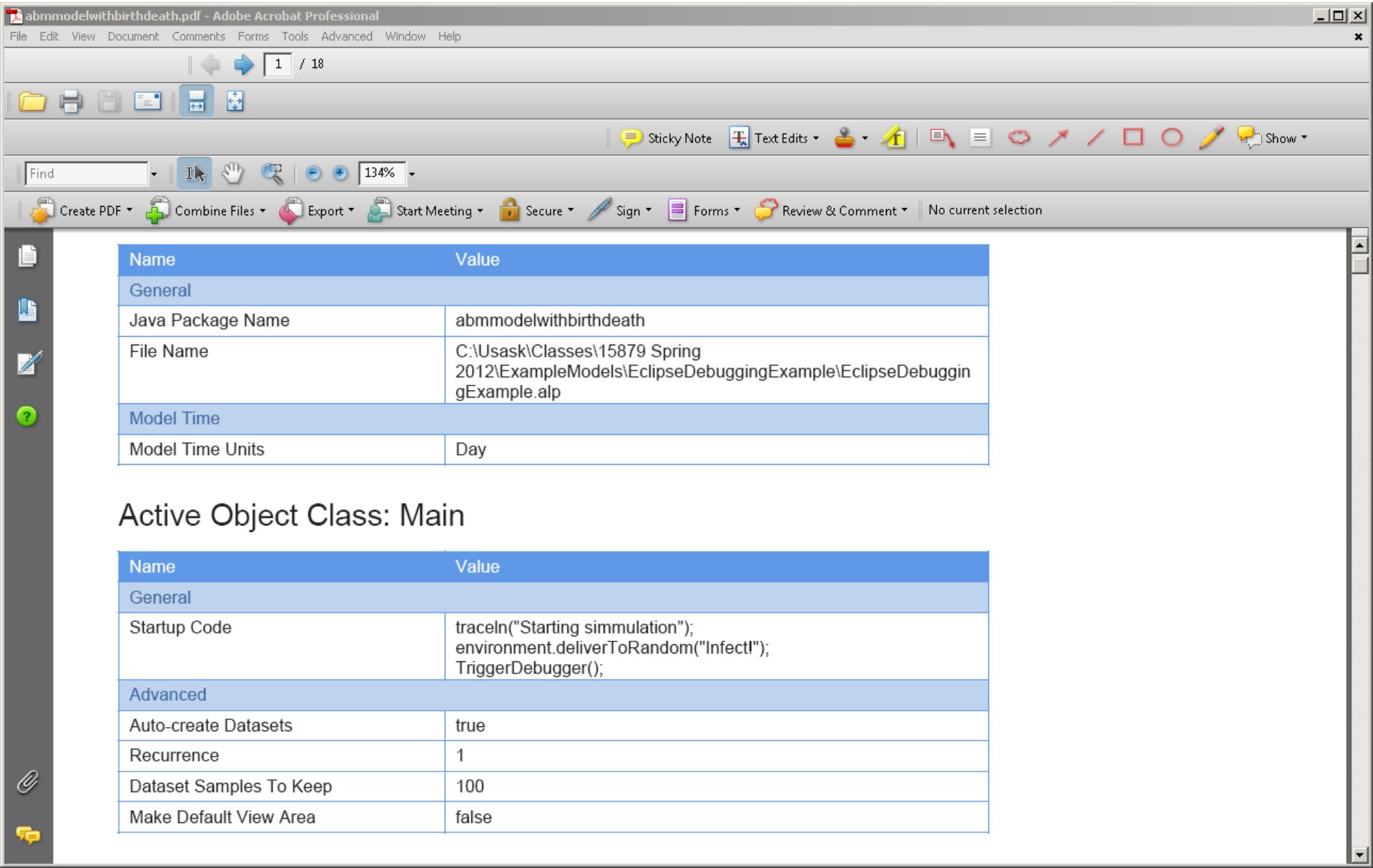
Generate Documentation



Selecting Documentation Output



Example Documentation



Name	Value
General	
Java Package Name	abmmodelwithbirthdeath
File Name	C:\Usask\Classes\15879 Spring 2012\ExampleModels\EclipseDebuggingExample\EclipseDebuggingExample.alp
Model Time	
Model Time Units	Day

Active Object Class: Main

Name	Value
General	

Startup Code	traceln("Starting simmulation");
	environment.deliverToRandom("Infect!");
	TriggerDebugger();

Advanced	
Auto-create Datasets	true
Recurrence	1
Dataset Samples To Keep	100
Make Default View Area	false

Incremental Delivery

Best Advice: Start Simple!

- It is easy to get lost in these models
- Focus on building up the models incrementally, as insights arise
- Innovate off of simple examples
- Avoid the temptation of the “big bang” project

Some Benefits of Incremental Delivery

- Morale: Get products soon
- Discover problems sooner
- Flexibility to change direction in way that reflects new knowledge & understanding
- Easier to estimate time required for next deliverable
- Can better handle slower progress or unexpected schedule limits: At least get some value from dev.
- Get more insight about what to do by tangibly working with a produced artifact
- Can avoid “gilding the lily” by heading off unnecessary development

Continuous Integration

Continuous Integration

- Continuous integration involves ongoing integration of different people's contributions to an underlying artifact
  - This is in contrast to the traditional "big bang" approach of integrating all elements at once
- Continuous integration is conceptually different from but helps support incremental delivery

Continuous Integration: Advantages

- Cooperation: Greatly reduces integration headaches
  - Reduced likelihood of merge conflicts
  - Easier, less wasteful to fix if conflict occurs
  - Allows bigger teams to function nimbly
- Quicker identification of problematic modifications & bugs
- Helps identify state of project via smoke tests, availability of executable
- Improved estimation, flexibility for shipping
- Feedback: Reduces need for status reports, polling
  - Automated build validation test (BVT) scripts
- Improves team morale
- Helps force fixing bugs before continuing

Managing Process Complexity

Process Complexity: A Barrier to Quality ABM Modeling

- Medium+ scale ABM projects generate a large # & diversity & versions of related artefacts
- Careful coordination of these artefacts is important for ensuring quality insights
- Efficient coordination is important for productivity
- Existing tools offer limited support for such coordination
- Difficulties limit what can be accomplished

Common Elements of the MP

- Creation of a modeling project
- Successive model versions are created for that project
- Each version is evaluated wrt a scenario set
  - Each scenario is motivated by some intention
  - This frequently includes a baseline and alternative scenarios
  - Frequently the set of scenarios exhibits some systematic structure
  - Results are analyzed (often in external docs)
- There is a frequent need to share access to these artifacts

Important Gaps in Software Support

- Model version control
  - Rollback
  - Comparison with earlier versions
- Ability to collaborate on shared artifacts
  - Communication of artifacts across machine/institutional boundaries
- Reification of structured scenario collections
- Lack of explicit links & referential integrity b/t
  - Versions & scenarios
  - Conceptually linked versions
  - Metadata & data
    - Motivation for creating scenario collection & scenario outputs
    - Artifacts & docs on intentions for producing them
    - Definition of scenario & output
    - Output & analysis documents
- Distributed evaluation of large scenario sets

Why the Gaps Matter

- Process transparency
- Risk of modeling errors
- Client confidence
- Speed of learning
- Modeling efficiency
- Practical limits on project scope

Risk-Driven Testing

Testing: Not Just "Finding Bugs"

- Identifying other quality problems
  - Design departures from requirements
  - Usability problems (particular power users)
- Should focus on *important bugs*
- Give immediate feedback on rough quality
  - Broad look at entire system
- Identify usability issues early thru test design
- Using different bug identifications than skills than developers
- Effective reporting critical

JUnit Tie-ins

- Tools like Junit can be used to do some testing against AnyLogic models
- Broad AnyLogic testing is made mor challenging by need to create appropriate test harnesses for testing extensions of AnyLogic classes
- Suggestion:
  - Create alternative experiments for focused testing
  - Create alternative startup logic in Main that calls testing-specific methods

Prototypes

Some suggestions on Prototypes

- Try adding in detail in experimental (throw-away) prototypes before commit to it
- Prototype two ways of approaching something
  - This takes time, but may save more time

Prototypes – Not Just for the UI!

- Engineering mockups critical in other domains (e.g. construction)
- Identify relationships between components
- Identify risks
- Identify potential engineering savings from design changes
- Understanding interfaces between components
- Understanding testing priorities

Prototypes

- Minimal mockups to test (grouped) ideas
  - Examine key issues w/o assumption that using this approach
- Risk analysis e.g.
  - Prototype most challenging or highest priority questions
  - Pick best idea from each affinity group for prototyping
  - Prototype each affinity group
- Should be for throw-away use – do not to use code
- Later use should be driven by open issues & decision making needs

Peer Reviews & Inspections

Reviews: Why?

- More cost-effective than testing
  - IBM found 3.5 hours/error for inspection removal vs. 15-25 hours/error for testing
- Easily pay for themselves ("Quality is Free")
- More flexible than testing
  - **Need not wait for executable code**
    - Can perform at all stages of software engineering process
    - Can be done early in the development of a component
  - Can assess communications issues (clarity, style, commenting, etc.)

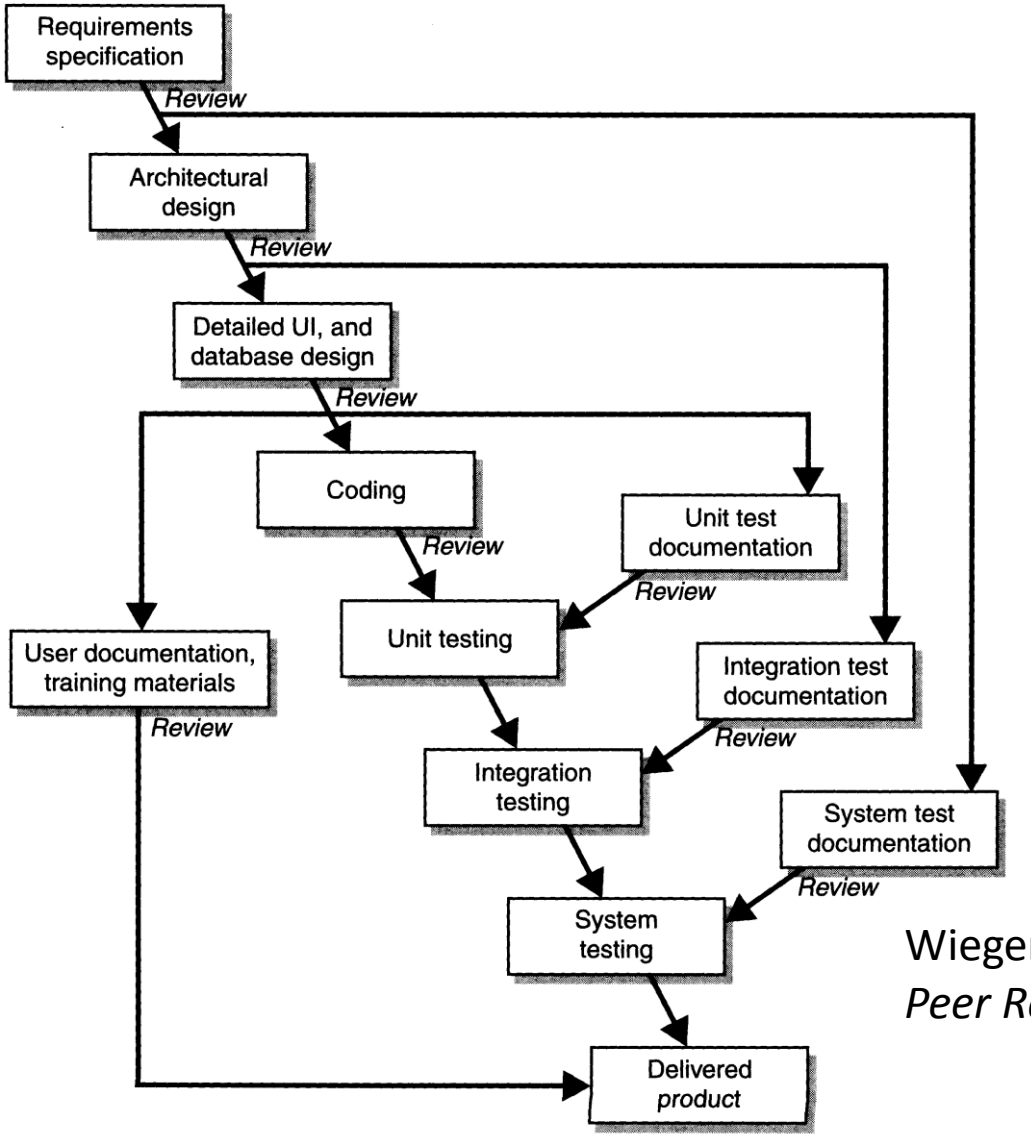
Importance of Early Reviews

- Requirements
- Early artifacts have disproportionate impact on development process
- Marketing documents
- UI design
- Design
- Unit implementation
- Unit testing

Other Benefits of Peer Reviews To…

- Person reviewing the artifact (Clarify understanding, learn coding tricks, stylistic ideas)
- Person whose artifact is being reviewed
  - Improving technique, learn
- Broader culture
  - Spread of knowledge about code base
  - Spread of knowledge of standards, coding styles
  - Code written with other people in mind

Good Points for Peer Reviews



Wiegers,
Peer Reviews in Software

Guidelines for reviews

- Keep impersonal: Focus on artifact, not people
- Keep review team small (3-7 participants)
- Try to identify -- **but not solve** -- problems during review
- Limit meeting to no more than 2 hours
- Require advanced preparation for **formal** reviews
- Be sensitive to cultural and human components
- Prioritize focus for more major issues

Inspection: Best Practices (Wiegers)

- Plan inspections to address project & inspection objectives
- Inspect upstream documents first
- Begin inspect documents early in their lives
- Check against source and related documents
- Prepare & inspect at your organization's optimum rates
- Focus on major defects
- Measure your benefits from inspections
- Emphasize defect prevention and process improvement
- Use serious, quantitative entry and exit conditions

Stages: Planning

- Participants review material on own before meeting
- Moderator assigned at this point
- Author contributes objectives for inspection
- Based on historic data moderator estimates # of meetings required to do reviews of desired scope
- Moderator
  - Invites participants
  - Helps author prepare package of materials for inspections
  - Distributes package to participants several days ahead of time

Stages: Overview

- Often a separate meeting
- Author more informally describes perspective on product
- Sometimes the inspection package is distributed during this meeting
- Sometimes skip if
  - Participants already familiar with product
  - Overview can be described in package

Stages: Preparation

- Most preparation centers around inspection package
  - The deliverable to be inspected
  - Standards/Requirements/Specifications
  - Typo list/individual issue log
  - Work aids to help identify defects
    - (e.g. Common defects for this sort of deliverable)
  - Test documentation to verify this deliverable

Stages: Meeting 1

- Deliverables
  - "inspection summary report“ (moderator)
    - Work product appraisal
    - Information to communicate to mgmt, etc.
  - "issues log“
  - Indication of what changes are needed to complete inspection process
- May stop inspection if identified errors are too serious to make it worth it to continue

Meeting Participant Roles

- Author (shares perspective)
- Moderator: leads process
- Reader: presents pieces of code (and perspectives on) to inspectors
  - Can help cataylze shared understanding by inspectors
- Inspectors: (any participant, including those assigned to other roles)
  - Can critique code
  - Can identify possible issues where errors
- Recorder: Documents issues
- Typically 3-4 participants

Stages: Rework

- Author addresses most items in issues log
- Sometimes issue log items get assigned to others
- Sometimes just log defects in defect control system to be followed up later
- Result
  - Updated work product
  - Annotated issue log indicating resolution for each item

Stages: Followup

- Often with moderator as "verifier" (moderator decides when process is over)
- Verifier confirms that changes have been successfully made
- Baselining of changed deliverable into SCCS

Stages: Causal analysis

- This basically uses inspection process to improve
  - The development process
  - The inspection process
- Focus on process improvement and not on people
- Try to identify root cause of defects
  - E.g. Ambiguous explanations in requirements, design specs, inconsistent naming conventions